AMENDMENTS TO THE CLAIMS

Claim 1 (Currently Amended) A garbage collection system ~~that frees~~ for freeing memory areas corresponding to objects that are no longer required in an execution procedure of an object-oriented program ~~composed of~~ including a plurality of threads, the garbage collection system comprising:

a selection unit operable to select the threads of the object-oriented program one at a time;

an examination unit operable to examine each ~~execute examination processing with respect to the~~ selected thread, the examination ~~processing~~ including ~~procedures of~~ (i) stopping execution of ~~the~~ a selected thread, (ii) finding an object that is accessible from the selected thread by referring to an object pointer of the selected thread, (iii) managing the ~~found~~ object found by the finding as a non-freeing target, and (iv) resuming execution of the selected thread;

a detection unit operable to~~, when having detected~~, after the selection unit has commenced selecting, (i) detect ~~that~~ an object pointer that has been processed as a processing target by a currently-executed thread, and (ii) manage an object ~~indicated~~ identified by the ~~processing target~~ object pointer detected as the processing target, as a non-freeing target; and

a freeing unit operable to, after the examination unit has ~~processing has been~~ completed examining ~~with respect to~~ all of the threads, free memory areas that correspond to objects other than the objects that are managed as non-freeing targets, wherein:

the detection unit performs the detection only when the currently-executed thread has not yet been subject to examination by the examination unit; and

the detection unit includes:

a finding sub-unit operable to, when the detection has been performed, store, to a working memory area that corresponds to the currently-executed thread, the object pointer detected as the processing target and an object pointer of an object that can be reached from the object pointer detected as the processing target; and

a management sub-unit operable to, while the execution of a selected thread is being stopped by the examination unit, manage, as a non-freeing target, an object that can be reached from the object pointer stored in the working memory area corresponding to the

currently-executed thread.

Claim 2 (Cancelled)

Claim 3 (Currently Amended) The garbage collection system of ~~Claim 2,~~ claim 1, wherein:

~~the examination processing is processing for,~~ when an object identified ~~indicated~~ by an object pointer in a stack corresponding to the selected thread is found to be accessible, the examination of the examination unit repeatedly performs ~~performing~~ a procedure, ~~of,~~ only when both (a) the accessible object is not already being managed as a non-freeing target and (b) an object pointer exists in the accessible object, of further finding that an object identified ~~indicated~~ by the object pointer in the accessible object is accessible;[[,]]

the selection unit, after a first selection, further performs selection if, after the examination of the first selection ~~processing~~ has been performed by the examination unit, any threads ~~out~~ of the plurality of threads remain that have not been subject to the examination of the examination unit ~~processing, and~~; and

the selection unit refers to information about the threads, and makes the selection based on one or more predetermined thread selection conditions.

Claim 4 (Currently Amended) The garbage collection system of Claim 3, wherein:

the thread selection conditions include a condition indicating that any threads ~~whose~~ having a thread state that is a wait state are to be selected before any threads having a ~~whose~~ thread state that is a state other than the wait state ~~, and~~; and

if a thread having a ~~whose~~ thread state that is the wait state exists when making the selection, the selection unit selects the thread having the ~~whose~~ state that is the wait state.

Claim 5 (Currently Amended) The garbage collection system of Claim 4, wherein the thread selection conditions include a condition indicating that any threads having a ~~whose~~ thread

priority level that is low are to be selected before any threads ~~whose~~ having a thread priority level that is high.

Claim 6 (Currently Amended) The garbage collection system of Claim 5, wherein the thread selection conditions include a condition indicating that any threads ~~whose~~ having a corresponding stack size that is small are to be selected before any threads ~~whose~~ having a corresponding stack size that is large.

Claim 7 (Currently Amended) The garbage collection system of Claim 6, further including a memory management mechanism that manages memory with use of a memory management unit ~~(MMU)~~,

wherein each time an object is to be generated, a memory area corresponding to the object is allocated by the memory management mechanism, and

wherein the freeing unit frees the memory areas via the memory management mechanism.

Claim 8 (Currently Amended) The garbage collection system of Claim 3, wherein the thread selection conditions include a condition indicating that any threads ~~whose~~ having a corresponding stack size that is small are to be selected before any threads ~~whose~~ having a corresponding stack size that is large.

Claim 9 (Currently Amended) The garbage collection system of Claim 3, wherein the thread selection conditions include a condition indicating that any threads ~~whose~~ having a thread priority level that is low are to be selected before any threads ~~whose~~ having a thread priority level that is high.

Claim 10 (Currently Amended) The garbage collection system of Claim 1, further including ~~using~~ a memory management mechanism that manages memory with use of a memory management unit ~~(MMU)~~,

wherein each time an object is to be generated, a memory area corresponding to the object is allocated by the memory management mechanism, and

wherein the freeing unit frees the memory areas via the memory management mechanism.

Claim 11 (Currently Amended) A garbage collection method that, in a computer, frees memory areas corresponding to objects that are no longer required in an execution procedure of an object-oriented program ~~composed of~~ including a plurality of threads, the garbage collection method comprising:

~~a selection step of~~ selecting the threads of the object-oriented program one at a time;

~~an examination step of executing examination processing with respect to the~~examining each selected thread, the examining ~~examination processing~~ including ~~procedures of~~ (i) stopping execution of ~~the~~ a selected thread, (ii) finding an object that is accessible from the selected thread by referring to an object pointer of the selected thread, (iii) managing the ~~found~~ object found by the finding as a non-freeing target, and (iv) resuming execution of the selected thread;

~~a detection thread of, when having detected,~~ after the ~~selection~~ selecting of the threads ~~in the selection step~~ has commenced, (i) detecting ~~that~~ an object pointer that has been processed as a processing target by a currently-executed thread, and (ii) managing ~~manage~~ an object identified ~~indicated~~ by the ~~processing target~~ object pointer detected as the processing target, as a non-freeing target; and

~~a freeing step of,~~ after the examining ~~examination processing~~ has ~~been~~ completed examining ~~with respect to~~ all of the threads, freeing memory areas that correspond to objects other than the objects that are managed as non-freeing targets, wherein:

the detecting is only performed when the currently-executed thread has not yet been subject to the examining; and

the detecting includes:

when the detecting has been performed, storing, to a working memory area that corresponds to the currently-executed thread, the object pointer detected as the processing target and an object pointer of an object that can be reached from the object pointer detected as

the processing target; and

while the execution of a selected thread is being stopped by the examining, managing, as a non-freeing target, an object that can be reached from the object pointer stored in the working memory area corresponding to the currently-executed thread.

Claim 12 (Currently Amended) The garbage collection method of Claim 11, wherein:

the computer uses a memory management mechanism that manages memory with use of a memory management unit ~~(MMU)~~, and, in an execution procedure of an object-oriented program, each time an object is to be generated, allocates a memory area corresponding to the object according to the memory management mechanism~~, and~~; and

the freeing ~~step~~ frees the memory areas via the memory management mechanism.

Claim 13 (Currently Amended) A computer-readable storage medium having a computer program stored thereon, the computer program for having a computer execute garbage collection processing that frees memory areas corresponding to objects that are no longer required in an execution procedure of an object-oriented program including ~~composed of~~ a plurality of threads, the computer program causing the computer to execute a method comprising: ~~garbage collection processing including:~~

~~a selection step of~~ selecting the threads of the object oriented-program one at a time;

~~an examination step of executing examination processing with respect to the~~ examining each selected thread, the examining ~~examination processing~~ including ~~procedures of~~ (i) stopping execution of ~~the~~ a selected thread, (ii) finding an object that is accessible from the selected thread by referring to an object pointer of the selected thread, (iii) managing the ~~found~~ object found by the finding as a non-freeing target, and (iv) resuming execution of the selected thread;

~~a detection thread of, when having detected,~~ after the ~~selection~~ selecting of the threads ~~in the selection step~~ has commenced, (i) detecting ~~that~~ an object pointer that has been processed as a processing target by a currently-executed thread, and (ii) managing ~~manage~~ an object identified ~~indicated~~ by the ~~processing target~~ object pointer detected as the processing target, as a non-

freeing target; and

~~a freeing step of,~~ after the examining ~~examination processing~~ has ~~been~~ completed examining ~~with respect to~~ all of the threads, freeing memory areas that correspond to objects other than the objects that are managed as non-freeing targets, wherein:

the detecting is only performed when the currently-executed thread has not yet been subject to the examining; and

the detecting includes:

when the detecting has been performed, storing, to a working memory area that corresponds to the currently-executed thread, the object pointer detected as the processing target and an object pointer in an object that can be reached from the object pointer detected as the processing target; and

while the execution of a selected thread is being stopped by the examining, managing, as a non-freeing target, an object that can be reached from the object pointer stored in the working memory area corresponding to the currently-executed thread.

Claim 14 (Currently Amended) The garbage collection method of Claim 13, wherein:

the computer uses a memory management mechanism that manages memory with use of a memory management unit ~~(MMU)~~, and, in an execution procedure of an object-oriented program, each time an object is to be generated, allocates a memory area corresponding to the object according to the memory management mechanism~~, and~~; and

the freeing step frees the memory areas via the memory management mechanism.

Claim 15 (New) The garbage collection system of claim 1, wherein the currently-executed thread is a thread that is being executed according to reference processing.

Claim 16 (New) The garbage collection system of claim 15, wherein the execution that is stopped by the execution unit is execution by an interpreter.